DESCRIPTION

TASK EXECUTION SYSTEM

TECHNICAL FIELD

The present invention relates to a task execution system including at least two pieces of processors.

BACKGROUND ARTS

There has hitherto been known a system having a function of changing a task to be processed (refer to, e.g., Patent document 1). In a multiprocessor system, however, if a certain processor falls into stoppage due to a fault, etc., it is impossible to assure an operation of the task processed so far by this processor, resulting in a problem that the operation of the whole system cannot be assured.

Note that in a system for actualizing a function in such a way that a plurality of tasks cooperate with each other through task-to-task communications, a task processing system (see, e.g., Patent document 2) capable of easily dealing with a change, etc. of a massage between the tasks, which is caused by addition/deletion, etc. of the task, is given as what is related to the present invention.

(Patent document 1) Japanese Patent Laid-Open Publication No.11-203149

(Patent document 2) Japanese Patent Laid-Open Publication No.6-95896

DISCLOSURE OF THE INVENTION

It is an object of the present invention to assure, even if a certain processor stops due to a fault, etc. in a multiprocessor system, an operation of a task processed so far by this processor and also assure an operation of the whole system.

The present invention, which was devised to accomplish the above object, is a task execution system including at least two processors, comprising a task management table registered with an associated relationship between at least a task, a main execution processor for executing the task and an in-charge-of-stoppage processor for executing the task when the main execution processor stops, means for selecting an executable task from among tasks registered in the task management table, means for checking, if a processor other than the processor trying to execute the selected task is registered as the main execution processor for the selected task, a stoppage state of the processor registered as the main execution processor, and means for executing the selected task if the processor registered as the main execution processor remains stopped.

According to the present invention, the task that is to be invariably executed by a certain processor (the main execution processor) is assigned beforehand to other processor (the in-charge-of-stoppage processor), and there is made a task accepting judgment of the task (including the task

assigned). Then, the task is executed by the pre-assigned processor when the above processor falls into the stoppage, thereby making it possible to actualize assurance of the operation of the task assigned and to assure the operation necessary for the system. As a result, there can be improved possibility of assuring the operation of the system even when the system is partially stopped.

Further, the present invention can be specified as follows.

A task execution system including at least two processors, comprises means for judging whether or not a task requested to be registered can be registered as a task of a main execution processor, means for judging whether or not the task requested to be registered can be registered as a task of an in-charge-of-stoppage processor, means for registering, if judged to be registerable as the task of the main execution processor and if judged to be registerable as a task of the in-charge-of-stoppage processor, an associated relationship between the task requested to be registered, the main execution processor and the in-charge-of-stoppage processor, means for selecting an executable task from among the registered tasks, means for checking, if a processor other than the processor trying to execute the selected task is registered as the main execution processor for the selected task, a stoppage state of the processor registered as the main execution processor, and means for executing the selected task if the processor registered as the main execution processor

remains stopped.

Moreover, the present invention can be specified by way of the invention of a method as below.

A task execution method in a task execution system including at least two processors, comprises selecting an executable task from among tasks registered in a task management table registered with an associated relationship between at least a task, a main execution processor for executing the task and an in-charge-of-stoppage processor for executing the task when the main execution processor stops, checking, if a processor other than the processor trying to execute the selected task is registered as the main execution processor for the selected task, a stoppage state of the processor registered as the main execution processor, and executing the selected task if the processor registered as the main execution processor remains stopped.

Still further, the present invention can be specified by way of the invention of a program as follows.

A program makes an information processing device including at least two processors, function as a task management table registered with an associated relationship between at least a task, a main execution processor for executing the task and an in-charge-of-stoppage processor for executing the task when the main execution processor stops, means for selecting an executable task from among tasks registered in the task management table, means for checking, if a processor other than the processor trying to execute the

selected task is registered as the main execution processor for the selected task, a stoppage state of the processor registered as the main execution processor, and means for executing the selected task if the processor registered as the main execution processor remains stopped.

Yet further, the present invention can be also specified as a storage medium stored with the above program, which can be read by an information processing device (computer).

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows an outline of an architecture of a task execution system by way of an embodiment of the present invention;

FIG. 2 is a flowchart for explaining an operation of the task execution system in the embodiment of the present invention; and

FIG. 3 is a flowchart for explaining the operation of the task execution system in the embodiment of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

A task management system will hereinafter be described with reference to the drawings by way of one embodiment of the present invention.

(Outline of Architecture of Present System)

FIG. 1 is a diagram for explaining an outline of an architecture of a task execution system.

(System Environment)

The task execution system in the present embodiment is actualized by a general type of information processing device 100 such as a PDA (Personal Digital Assistant), a personal computer and so on.

As shown in FIG. 1, the information processing device 100 includes two pieces of processors 110, 120 (of which one processor will hereinafter be referred to as a main execution processor 110, and the other processor will be termed an in-charge-of-stoppage processor 120 for the explanatory convenience's sake), a storage device 130 such as a hard disk device, etc., a memory 140 and so forth. Further, the information processing device 100 includes, in some cases, an input device (for example, a key set) for inputting various pieces of information and commands, an image display device (e.g., a liquid crystal display) for displaying a result of processing thereof, a voice output device (for instance, a loudspeaker), etc. (none of these devices are illustrated). Note that the two processors 110, 120 are exemplified for the explanatory convenience's sake, however, the present invention is not limited the two processors. For example, the present invention can be similarly applied even when three or more pieces of processors are provided.

(Task and Operating System)

A task 141, which is generally called a process or a thread, is a generic name of an execution unit of a program. The task 141 may be generated when in a task registration and

may also be previously generated (pooled) as done in the preceding application by the present applicant. The task 141 is, for example, a QoS (Quality of Service) task defined as a variable task capable of controlling a required quantity of resources.

The operating system (OS) 142 is, for instance, a real time OS having a function (as a scheduler) of scheduling the respective tasks 141 by a DM (Deadline Monotonic) method. Among the executable tasks 141 (corresponding to execution target tasks according to the present invention) at each scheduling timing, the task 141 exhibiting the shortest period of deadline time is set as an active task. Each of the tasks 141 is managed based on a task management table 143.

The task management table 143 is illustrated in the lowest part in FIG. 1. The task management table 143 is a table for managing pieces of information about the respective tasks, and is registered with pieces of the information about the tasks 141, such as a task ID 143a, a main execution processor ID 143b, an in-charge-of-stoppage processor ID 143c, a task execution parameter 143d, and so on.

The task ID 143a serves to identify each task 141.

The main execution processor ID 143b and the in-charge-of-stoppage processor ID 143c serve to identify the respective processors. Start timing, execution alignment time, deadline time, etc. are given as the task execution parameter 143d. The start timing is a period of time (period) till next execution start timing since execution start timing of each

task 141. When a certain task 141 is executed, it does not happen that this task 141 is executed afresh during this period. The execution assignment time is defined as a resource quantity (which is, for instance, a period of usage time of each processor) assigned to each of the tasks 141. It is to be noted that the resource is not necessarily assigned continuously for a period of assignment time throughout to the task 141 that has been once assigned the resource. A time assignment may be effected separately any number of times. Further, if preempted by a different task 141 having a higher priority than a certain task 141, processing of this task 141 is interrupted.

If the alignment time elapses within a certain period of time, nothing affects the system. This period of time is the deadline time. In the present embodiment, the task 141 exhibiting a short period of deadline time is given a higher priority than that of the task 141 showing a long period of deadline time. The task 141 having the short deadline time (i.e., the task 141 given the high priority) is set as an active task.

Predetermined programs such as API (Application Program Interface), etc. for providing the schedule function described above and other various functions that will be described later on, are read by the aforementioned information processing device 100 and installed into the operating system 142, thereby actualizing these functions. Note that the operating system 142 and the predetermined programs, etc. are pre-

installed into the storage device 120 or the like, and are properly read into the memory 140 and executed as the necessity arises (see FIG. 1).

(Operation at Task Registration)

Next, an operation of the task management system having the architecture described above will be explained with reference to the drawings. To start with, processes when registering the task will be described. FIG. 2 is an explanatory flowchart of the processes when registering the task.

The operating system 142, etc., is read and executed by the information processing device 100, thereby actualizing the following processes. When a task registration request is given from the operating system 142, a predetermined application, etc. (S100), it is judged whether or not the task 141 requested to be registered (which will hereinafter be also called a registration target task) can be registered as a task of the main execution processor 110 (which is identified, if the registration request in S100 contains the main execution processor ID, by this main execution processor ID) (S101). This is, in the case of registering, for example, the registration target task 141 as a task of the main execution processor 110, the judgment as to whether or not the execution can be done while keeping QoS in a way that includes this registration target task 141, wherein it is judged whether predetermined conditions are met or not.

As a result, if the registration target task 141 is

judged not to be the task of the main execution processor 110 (S101: No), this registration target task 141 is not registered, and registration unpermitted notification is given (S102).

While on the other hand, if judged to be registerable as the task of the main execution processor 110 (S101: Yes), it is further judged whether or not the task registration request (S100) is for only the main execution processor (S103). This can be, it is considered, judged by knowing whether or not, for example, the task registration request (S100) contains only the main execution processor ID (i.e., whether this request contains also the in-charge-of-stoppage processor ID or not).

As a result, if not judged to contain only the main execution processor ID (for example, if judged to contain also the in-charge—of-stoppage processor ID) (S103: No), it is further judged whether or not this registration target task 141 can be registered as a task of the in-charge-of-stoppage processor (S104). This is, for instance, if this registration target task 141 is registered as the task of the in-charge-of-stoppage processor 120 (which is identified, for example, when the registration request in S100 contains the in-charge-of-stoppage processor ID, by this in-charge-of-stoppage processor ID), a judgment as to whether or not the execution can be done while keeping QoS in a way that includes this registration target task 141, wherein it is judged whether the predetermined conditions are met or not.

As a result, if not judged to be registerable as the task of the in-charge-of-stoppage processor 120 (S104: No), this registration target task 141 is not registered, and the registration unpermitted notification is given (S102).

While on the other hand, if judged to be registerable as the task of the in-charge-of-stoppage processor 120 (S104: Yes), the registration target task 141 is registered in the task management tables of the main execution processor 110 and of the in-charge-of-stoppage processor 120 (S105). Namely, each task management table 143 is registered with the task ID 143a of the registration target task 141, the main processor ID 143b, the in-charge-of-stoppage processor ID 143c and the task execution parameter 143d.

While on the other hand, as a result of the judgment in S103, when judging that the registration request is for only the main execution processor (for instance, if judged not to contain the in-charge-of-stoppage processor ID) (S103: Yes), this registration target task 141 is registered in the task management table 143 of the main execution processor (S106). Namely, the task management table 143 of the main execution processor is registered with the task ID 143a of the task 141 requested to be registered, the main processor ID 143b, and the task execution parameter 143d.

As discussed above, when registering the task, there are assigned the processor ID (the main execution processor ID) of the processor that mainly executes the task and the processor ID (the in-charge-of-stoppage processor ID) of the processor

that executes the task in case of stoppage of the processor that mainly executes the task. Then, when the task registration request is given (S100), at first, the system making an acceptance judgment executes an acceptance process (S101, S103, S104), and, if executable, the task requested is registered (S105, S106).

(Operation at Task Switchover)

Next, processes when switching over the task will be explained. FIG. 3 is an explanatory flowchart showing the processes when switching over the task.

The operating system 142 (such as the scheduler), etc. is read and executed by the information processing device 100, thereby actualizing the following processes.

When the scheduler is executed (S200), an executable task is selected from among the registered tasks of the self-processor (S201). For instance, when switchover timing based on the task scheduling is reached, a task exhibiting a shorter period of deadline time is selected from among the tasks 141 registered in the task management table 143 of the self-processor. Note that if none of the executable tasks exist (S202 : No), the task switchover process is not executed but is terminated.

Whereas if the executable task exists (S202: Yes), it is judged whether or not the self-processor is set as the main execution processor for the task (which will hereinafter be also called a selected task) selected in S201 (S203). The task management table 143 of the self-processor is referred to

for making this judgment. The task management table 143 is registered with an associated relationship between the task ID 143a and the main execution processor ID 143b (see FIG. 1). It is therefore possible by referring to this task management table 143 to judge whether or not the self-processor is set as the main execution processor for the selected task.

As a result, if the self-processor is judged to be set as the main execution processor for the selected task (S203: Yes), the selected task is set as an execution task (S204). Namely, the selected task is executed.

While on the other hand, if the self-processor is not judged to be set as the main execution processor for the selected task (for instance, if a processor other than the self-processor is set as the main execution processor) (S203: No), it is further judged whether or not the processor set as the main execution processor for the selected task is stopped (i.e., a stoppage state of the processor set as the main execution processor is checked) and whether or not the self-processor is set as the in-charge-of-stoppage processor for the selected task (S205). The task management table 143 of the self-processor is referred to for making the latter judgment. The task management table 143 is registered with an associated relationship between the task ID 143a and the in-charge-of-stoppage processor ID 143c (see FIG. 1). It is therefore feasible by referring to this task management table 143 to judge whether or not the self-processor is set as the in-charge-of-stoppage processor for the selected task.

Incidentally, it is considered as a method for the latter judgment to check the stoppage state by directly querying, about an operation state, the processor set as the main execution processor for the selected task.

As a result, when judging that the processor set as the main execution processor for the selected task remains stopped (such as a case of falling into becoming inoperable due to a fault, etc.), and when the self-processor is set as the in-charge-of-stoppage processor for the selected task (S205: Yes), the selected task is set as the execution task (S204). Namely, the selected task is executed.

On the other hand, as a result of the judgment in S205, if it is not judged that the processor set as the main execution processor for the selected task remains stopped, or if the self-processor is not set as the in-charge-of-stoppage processor (S205: No), the operation returns to S201, wherein the processes from S201 onward are re-executed.

As discussed above, when each of the processors 110 and 120 executes the task, the task is executed based on the management information of the registered tasks. On this occasion, however, if it proves from reading the processor ID that a processor other than the self-processor is set as the main execution processor (S203: No), a stoppage condition of that processor is checked (S205). If stopped (S205: Yes), the task thereof is executed (S204).

Thus, the stoppage condition of the processor is checked at the operation timing of each task (S205), and hence it

follows that surrogation of executing the task is promptly conducted. Further, the task that is to be invariably executed by the main execution processor 110 is assigned beforehand to the in-charge-of-stoppage processor 120, whereby the task can be executed by the pre-assigned processor 120 when the processor 120 falls into the stoppage. Accordingly, it is possible to improve the possibility of assuring the operation of the system even when the system is partially stopped.

The present invention can be embodied in a variety of forms without deviating from the sprit or the principal features thereof. Hence, the embodiment discussed above is nothing but a mere exemplification in every aspect. The present invention should not be limitedly construed by description of the embodiment.

INDUSTRIAL APPLICABILITY

According to the present invention, the task that is to be invariably executed by a certain processor (the main execution processor) is assigned beforehand to other processor (the in-charge-of-stoppage processor). The task is executed by the pre-assigned processor when the above processor falls into the stoppage, thereby enabling the improvement of the possibility of assuring the operation of the system even when the system is partially stopped (when the main execution processor stops).